Unsupervised Clustering for Google Searches of Celebrity Images

Alex Holub*, Pierre Moreels*, Pietro Perona
California Institute of Technology, Pasadena, CA. 91125
holub@vision.caltech.edu, pmoreels@vision.caltech.edu, perona@vision.caltech.edu
* These authors contributed equally in this work

Abstract

How do we identify images of the same person in photo albums? How can we find images of a particular celebrity using web image search engines? These types of tasks require solving numerous challenging issues in computer vision including: detecting whether an image contains a face, maintaining robustness to lighting, pose, occlusion, scale, and image quality, and using appropriate distance metrics to identify and compare different faces. In this paper we present a complete system which yields good performance on challenging tasks involving face recognition including image retrieval, unsupervised clustering of faces, and increasing precision of 'Google Image' searches. All tasks use highly variable real data obtained from raw image searches on the web.

1. Introduction

The most common image searches involve celebrities [1]. Most major search engines currently index images based on keywords associated with the images and do not utilize visual information within the images. Using visual information for celebrity searches has the potential to drastically increase the precision of such queries and could also benefit related tasks such as finding all images of a particular individual within a personal photo album (e.g. asking for all images of Grandpa in your personal photo album).

The high variability inherent in images returned by typical image search queries, usually results in relatively poor performance with traditional face recognition algorithms. Many current algorithms involving face data are tested using controlled data-sets, for instance the CMU PIE data-set [6] or Yale face data-set, and often do not perform well or have not been applied to data exhibiting real-world variability. Notable exceptions include the work of Zisserman et al (see, for instance [3, 4, 9]) who has worked extensively with both real video and images, as well as the work of Berg et al. [2] who use both text and images to automatically associate names to faces from news articles.

[1] Internal communication with AOL.

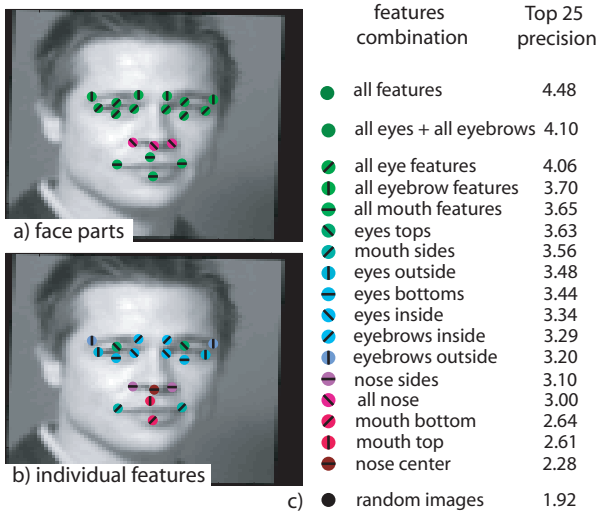

Figure 1. Which facial key-points are most useful? We rank the ability of different individual facial key-points to recall images of the same person within a large image data-set (see Section 5 for a more thorough description of the performance metric) (a) The performance of different features on recall experiments. Corresponding number and color-code in panel c) indicates the precision within the top 25 images, higher number is better. Recall performed only with: eyebrows structure, eyes structure, nose structure and mouth structure. (b) Performance when a face is characterized by a single individual patch (two in case of symmetry, e.g. both sides of the mouth are included together). The same color scale is used for panel a) and b). (c) Scores of parts and individual features. The set consisting of all parts performs best, followed by the eyes and eyebrows structures. Overall all features related to the eyes and eyebrows perform well, but best performance is achieved by using all features. 'Random images' is the baseline method that draws randomly 25 images and indicates the precision. This experiment used 7×7 patches and raw intensity values, the ranking did not change significantly when using 13×13 patches and image gradients. Note that we used an L1 distance to measure similarity and did not apply our learned distance metric for this comparison.

978-1-4244-2154-1/08/$25.00 ©2008 IEEE

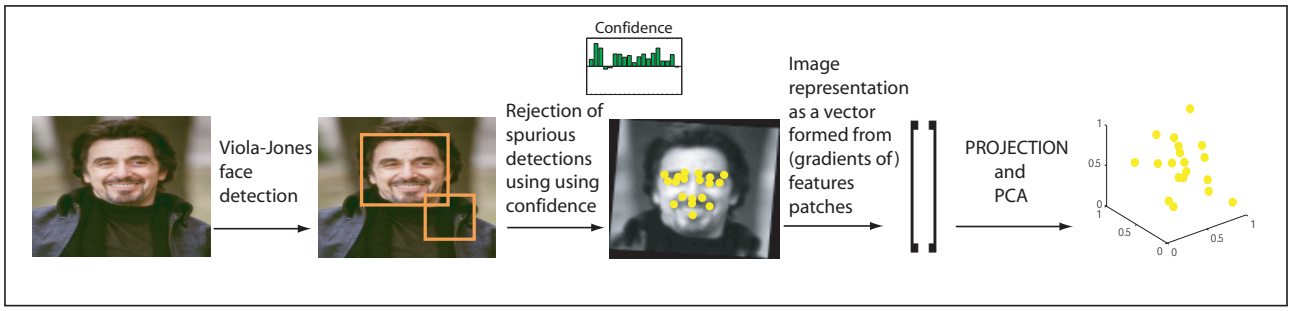

Figure 2. Steps used to transform a raw image into a feature vector.

In this work we develop a system capable of accommodating the rich variability inherent in real world images. In particular we utilize the Viola and Jones face detector [10] to identify the location of potential faces in images. Furthermore, we develop methods for learning distance metrics between faces in order to appropriately weight, shear, and scale the facial representations. Related work involving learning distance metrics include [11, 12] and the well-known linear discriminant-based FisherFaces [1] which use a combination of Eigenfaces and Linear Discriminant analysis to learn an appropriate linear mapping for face representations. In our last set of experiments we show how our system can be used to automatically cluster images of the same person retrieved from Google image searches.

Section 2 describes the general algorithm we employ. In Section 4 we describe the face detector, the facial feature finder, and facial feature representations used. In Section 5 we show how to create an effective distance metric. Section 6 and 7 shows and compares results of our complete system. We conclude in Section 8.

2. Performance Metrics

In this section we describe typical tasks involving face retrieval and the corresponding performance metrics we use to evaluate the performance of our system. The three primary image performance metrics we consider are:

(1) Given a query image of a particular person, how often is the nearest neighbor to this image, the same individual? This task would be useful in such applications as finding the most similar-looking celebrity to a person. The rank of the first occurrence of the queried individual among nearest neighbors, is termed *Best Rank Distance*.

(2) Given the K nearest neighbors to a target individual (K is arbitrary), how many images are of this individual? Think of, for instance, a Google Image Search - the first page of results contains, say, 25 images. We would like a high number of these images to be of our target individual. This is the detection performance with a recall of 25, we term it *Top 25 Precision*. Note that in our experiments we limit the number of images of each single individual to 10, such that the highest value achievable by *Top 25 Precision* would be 9.

(3) The final metric computes the distances from a query image to all images (same individual or not). From this list, we extract the ranks of all images of the query individual, and take the average of these ranks. This process is repeated for all images of this individual, the 'average of the averages' is then computed and called *Cluster Distance*. It characterizes how far the set of all images of the target individual is to the query image. In other words, it characterizes how tight the cluster of images of the target individual is. The Cluster Distance is normalized by the total number of images.

3. Data-Sets

Here we describe the data-sets used for most of the experiments described in this paper. In creating our data-sets we used the web and thus our data-set suffered from, among others: large variations in lighting, no manual alignment, and varied resolutions (our resolution varied from about 100×130 to about 500×800). We collected a data-set of 99 individuals by typing in celebrity names as queries into popular search engines and collecting all images which contained the celebrity. We used celebrities as they tended to have a large number of images available. We call this data-set the 99-Celebrity data-set. We ensured that each individual had a minimum of 10 images. We also collected a set of 200 generic face images (see Table 1) for a description of the size of the data-sets. Figure 3 shows the typical variability of our faces.

4. Feature Extraction and Representation

In this section we describe how, given an image containing a face, we extract a feature representation which is somewhat robust to the high variability inherent in natural images. Figure 2 shows a schematic of our system. Note that the resulting feature representations should reflect fa-

Figure 3. Two examples for the 'Top 25 Precision' retrieval task. We queried for Will Smith (panel a) and for Owen Wilson (panel b) - the query images are showed in the top left with a blue outline. In both cases the query returned 7 correct results (green outline) out of the 25 closest matches after projection and PCA. It is important to remember that the category 'Will Smith' and the category 'Owen Wilson' contain only 10 examples each, therefore a perfect answer would return 10 samples from the target category out of 25 results.

cial identity, i.e. images of the same person will be closer to one another than images of different people.

In order to detect faces we use the OpenCV [5] implementation of the face detector from [10]. The output from the face detector is a set of bounding boxes which identify faces positions in images. Most of the 1266 images in our '99-Celebrities' data-set (see Section 3) yield a single detection, however in 68 cases it mistakenly finds two or more faces. Next, these bounding boxes are used as input for the Everingham facial-feature detectors [3]. This detector identifies the position of 19 features facial features as well as the confidence it has with these detections. The facial-feature identified are shown in Figure 5.

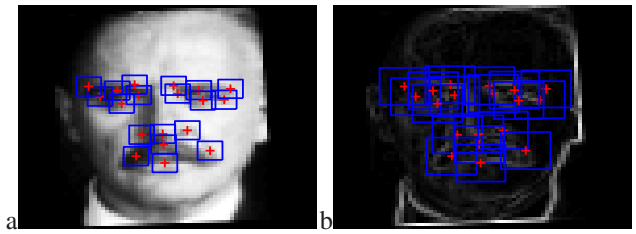

Figure 4. Example of the relative size of each patch. (a) Examples of 7×7 patches on a raw intensity figure (this yielded the optimal performance for raw patch representations, see Figure 6). (b) Example of gradient image and the size of extracted patches, $13x13$ patches performed best when using the gradient as a feature.

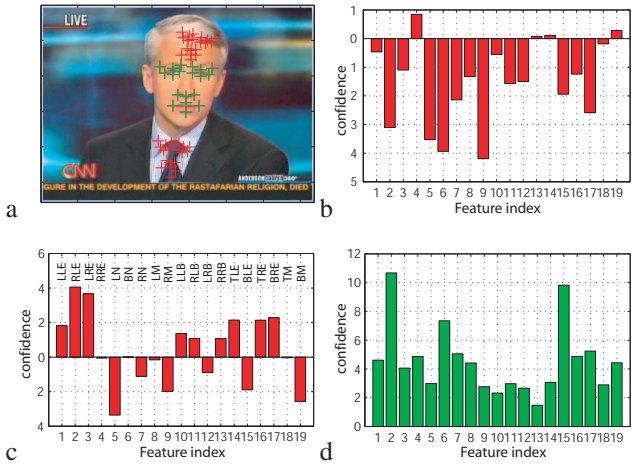

Figure 5. Example of Viola and Jones detection and Everingham feature detectors. (a) The Viola-Jones face detector found 3 'faces' in the image. b-c-d) confidences associated to each feature from each of the 3 detections. The following abbreviations are used: L=left, R=right, T=top, B=bottom, B=eyebrow, E=eye, N=nose, M=mouth. Using our heuristic based on parts' confidence, the incorrect detections are rejected (in red - the detection on the forehead corresponds to confidence scores in b., the detection on the tie corresponds to confidence scores in c.). The correct face is accepted (shown in green)

We used the following heuristic to discard spurious face detections: let CF the 19-dimension confidence vector for a face, $C^+ = max(CF, 0)$ its positive component and $C^- = -min(CF, 0)$ its negative component. We accept a face detection if $\sum_{k=1}^{19} C^+ > 4 * \sum_{k=1}^{19} C^-$, and reject it otherwise. We successfully rejected all but 4 false face detections, and introduced only 4 new false rejections. Figure 5 shows an example of successful rejection of spurious matches on an image that generated 3 detections.

Image Set	Total #img.	VJ Mult. det.	False al.
99 Celebs	1066	68	4
BG Celebs	200	15	0

Table 1. Table showing the size of our data-sets as well as the performance of the Viola and Jones detections. (First Column) The two data-sets, both a set of 99 individual celebrities downloaded from the web and a set of 200 other images also downloaded from the web. (Second Column) The total number of images in each data-set. (Third Column) Total number of images for which the Viola and Jones algorithm generated multiple detections. (Last Column) Number of remaining false alarms after heuristic based to remove spurious detections (see Section 4).

4.1. Facial Feature Representation and Size

For each detection accepted by the previous steps, we normalize the face bounding box to a fixed size of 80×80 pixels and rectify variations in orientation by aligning the eye corners to the same position for all images. We characterize each feature in a face by a patch of variable size extracted around the feature location. The set of patches extracted for a face are concatenated into a long vector that forms a representation of this face.

The choice of patch size for a feature representation is a trade-off: If patches are too small, the representation is sensitive to artifacts in the image, and not discriminative enough as the patch fails to capture enough of the local texture around the feature. Conversely, if the patches are too large, features include too much detail specific to a particular image and have poor generalization properties (see Figure 4). In this section, we investigate the influence of the patch size on the recognition performance.

For this Section and Section-4.2 only, for the sake of speed, we did not optimize the distance metric used nor reduce dimensionality with PCA. Rather, we used the commonly used L_1 and L_2 distances between patches. As a consequence, the performance reported in this experiment is lower than the results in Section 6. Figure 6 describes the results of our experiments. The performance of patch sizes from 3×3 pixels up to 23×23 was computed for the three score measures defined in Section 2. We computed the score with L_1 and L_2 distances, both when sampling patches from the raw intensity image and when sampling them from the gradient image (gradient provides some invariance to lighting conditions). Overall, the best patch sizes when using raw image intensity were 7×7 and 9×9 pixels, while larger patches performed better when using gradients (13×13 and 15×15 performed best). Since shorter representations are preferable for search purposes (due to the curse of dimensionality), we used raw intensity and 7×7 patches in further experiments.

4.2. Evaluation of face subparts

Which features in the face are most important for recognizing a specific person? In an attempt to answer this question and further optimize our system, we investigated the performance of various subparts of the face on a simplified experiment (see Figure 1). Features were extracted using 7×7 patches and intensity values. Faces were characterized by various combinations of features. One experiment focused on the sets of features that form face parts: eyebrows, eyes, nose and mouth. The other experiment focused on individual features. For features which occur symmetrically on both sides of the face, both left and right feature were included together.

Figure 1 shows the performance for the 'Top 25 detection' (see Section 2) criterion, color-coded by decreasing performance. Panel and c) indicates that as expected, face structures perform better than individual features. In particular, the most successful face structures are the eyebrows and the eyes. This was the case both when considering face parts and when considering individual features. Interestingly, this is consistent with the human performance results from [8], where Sinha reports that eyebrows are among the most important features for the human face recognition task.

5. Learning a Distance Metric

In the previous section we described how to automatically detect a face and extract features of the face. We now suggest that each component of our feature vector should not be weighted equally, i.e. certain parts of the face may be more important for recognition than other (Figure 1 suggests this is a reasonably intuition as different facial features are better and worse for recognition). How should we weight these features? We proceed by learning a metric which increases performance on recognition tasks.

Our goal is to learn a distance metric between any two images such that images of the same person are deemed close to one another, while images of different people are farther from one another. More formally consider each celebrity indexed by c. The feature representation for an image i in class c is denoted by $\vec{x}_i^c$. The following cost function follows naturally from the criteria just mentioned:

$$\mathcal{C} = \sum_c \sum_{i,j \in c} \sum_{k \notin c} \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k) \qquad (1)$$

By minimizing this cost function, we want to enforce that pairs (i, j) of images within a same individual or class c have a lower distance than pairs of images (i, k) taken from different individuals. In Equation 1, we want $\text{Dist}(\vec{x}_i, \vec{x}_j)$ to be as small as possible, and $\text{Dist}(\vec{x}_i, \vec{x}_k)$ to be as large as possible. With a given set of images of various individuals, we can form triplets (i, j, k) to use as a ground truth training set in Equation 1, to optimize the distance function Dist.

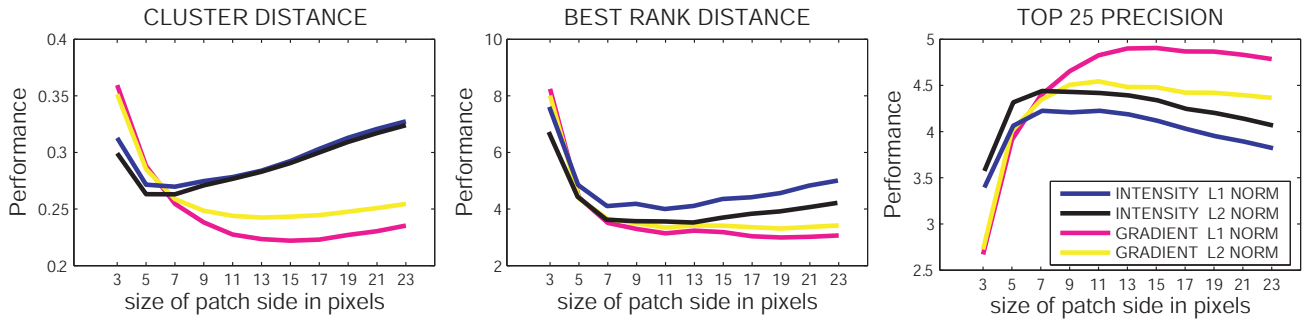

Figure 6. Variation of our three performance criteria with size of the patches used for face representation. We plot results for a simplified experiment for which no projection was performed, L_1 and L_2 distances were used on the raw patches. We display results of the simplified experiment both using image intensities and image gradients. X-axis: the size of patches extracted. Y-axis: performance using various metrics. All experiments averaged over the set of 99 Celebrities (see Section 6 for more details). (Left) Comparison using *Average Rank Distance* metric. (Center) Comparison using *Best Rank Distance*. (Right) Comparison using *Top 25 Precision*. The best performance is consistently obtained when using 7×7 patches in the case of raw intensity, and 13×13 patches when using image gradients.

Let us consider the distance $\text{Dist}(\vec{x}_i, \vec{x}_j)$. We now replace $\vec{x}_i$ by its mapping $\vec{y}_i = \phi(\vec{x}_i)$ by a kernel function ϕ. We can then re-write our distance function as $\text{Dist}(\phi(\vec{x}_i), \phi(\vec{x}_j))$. The L_1 and L_2 distances between mapped feature vectors are written as follows:

$$L_1 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sum_{r=1}^{Q} \left| \phi(\vec{x}_i) - \phi(\vec{x}_j) \right| \quad (2)$$

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sqrt{\sum_{r=1}^{Q} \left(\phi(\vec{x}_i) \phi(\vec{x}_j) \right)^2} \quad (3)$$

where Q is the dimensionality of the target space of ϕ. ϕ can be an arbitrary function. However, if we consider ϕ to be a linear function (i.e. a matrix Φ) and we let $M = \Phi\Phi^t$, then we can re-write the squared L_2 distance as:

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \quad (4)$$

Now consider again Equation 1. We would like to minimize this function. We proceed by using the conjugate gradient method. To use it, we need the derivatives of the cost function w.r.t. Φ for the L_1 and L_2 distances.

For the L_2 distance, we consider the simpler task of computing derivatives of the squared cost function w.r.t. $M = \Phi\Phi^t$:

$$\frac{\partial}{\partial M} (Dist^2_{L_2}(\phi(\vec{x}_i), \phi(\vec{x}_j))) \quad (5)$$

$$= \frac{\partial}{\partial M} \left((\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \right) \quad (6)$$

$$= (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \quad (7)$$

For the L_1 distance, the component (p_0, q_0) of the derivative is

$$\frac{\partial}{\partial \Phi_{p_0 q_0}} \sum_p \Big| \sum_q \Phi_{pq} a_q \Big| = \frac{\partial}{\partial \Phi_{p_0 q_0}} \Big| \sum_q \Phi_{p_0 q} a_q \Big| \quad (8)$$

$$= a_{q_0} \cdot sign(\sum_q \Phi_{p_0 q} a_q) = a_{q_0} \cdot sign((Ma)_{p_0}) \quad (9)$$

where we used the simplifying notation $a = x_i - x_j$. This can be written as the product of two vectors:

$$\frac{\partial}{\partial M} \left(Dist_{L_1}(Mx_i - Mx_j) \right) = \begin{pmatrix} sign(y_1) \\ . \\ . \\ . \\ sign(y_Q) \end{pmatrix} (x_i - x_j)^t \quad (10)$$

where $y = M(x_i - x_j) = Ma$. Note that there is a measure zero set when the derivative is undefined, namely when $\Phi(\vec{x}_i) = \Phi(\vec{x}_j)$. If all feature vectors $\vec{x}$ are unique this can only be satisfied when Φ is not full rank. We never encountered this situation in practice.

Finally consider again Equation 1. We may want to vary how much we penalize the difference, $\mathcal{U} = \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k)$, when i and j do not belong to the same class or when i and k do belong to the same class. For instance if we are interested in the *Closest Rank Image* we may not want to penalize heavily inequalities which result in large positive values of U, while, if we are interested in the *Average Rank Distance* we would penalize large positive values of U. We include a regularization term into our cost function with this desired behavior: $\mathcal{C}_{reg}(x) = e^{\frac{x}{\alpha}}$. Increasing the value of α reduces the influence of large positive values of U while decreasing α increases the influence of large U values. We ran experiments using various values of α shown in Figure 7. The total cost function is optimized using the conjugate gradient algorithm and usually converges after 100 iterations. We restart the algorithm multiple times ($3\times$) to avoid local minima.

6. Results

We evaluate the performance of our complete system on the data-sets described in Section 3. We use the three perfor-

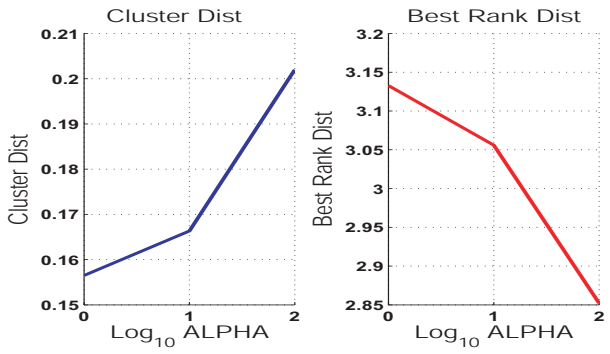

Figure 7. The effect of varying the steepness of the slope for our regularization term $\mathcal{C}_{reg}$ on two different performance metrics. (Left) Effects on Average Rank Distance Metric. Note the decrease in performance as we increase α. (Right) Effects on Best Rank Distance. Note that performance *increases* as we increase α. Increasing α results in the optimization giving equal weight to incorrect relative distances which are very far from one another and very close to one another. This intuition is consistent with the results shown in the plot. Results shown from using an L_1 metric on 7×7 intensity patches.

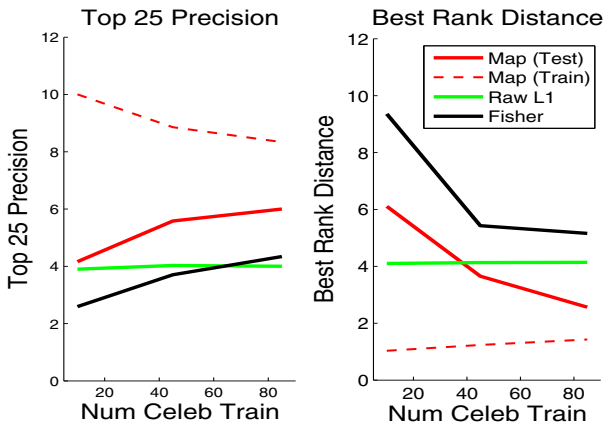

Figure 8. Effects of the number of individuals (celebrities) used for training on performance. (Left) Top 25 Precision metric. Results averaged over all celebrities in the test set. An L1 distance metric is optimized using 7×7 intensity patches. The x-axis is the number of individuals used for training ($\times 10$ this number of images are used for training). The y-axis is the precision of the top 25 returned celebrities. Note that the maximum achievable value is 9. Green line is the raw L1 performance of these features *before* mapping. Black line is the performance of FisherFaces [1] when trained using the same number of celebrities. Dotted red line is the performance on the train set of individuals. Solid red line is the performance on the test set of individuals. Note that we are over-fitting: we expect the solid and dotted red lines to converge when we are not over-fitting. Our distance metric is outperforming both the raw L1 distance as well as FisherFaces when we train with 85 individuals. (Right) Same as left plot but using the Best Rank Performance metric. Lower is better. Again we see over-fitting, but our distance metric still far outperforms the baseline methods despite over-fitting. In this case best performance would be 1, which occurs when identical celebrities would always be neighboring one another. Note that we separate our training set of celebrities used to construct the mapping from the test set of celebrities before each run of the experiment.

mance criteria described in Section 2. We did not perform any pre-processing on these images.

We conducted numerous experiments using our learned distance metrics. We compared this performance to both the raw feature representations as well as FisherFaces. In all experiments we initially projected our features down to 100 PCA dimensions, and our mapped feature space always contained 50 dimensions. I.e. Φ was a matrix of dimensionality 50×100. We use $\alpha = 1$ for these experiments.

In Figure 8 we varied the number of celebrities we train with in order to understand the asymptotic properties of learning with our distance metric. These plots have two main take-home messages: (1) Our learned distance metric performs well when compared to using either the raw features or FisherFaces (both techniques are widely used in the literature). (2) We are over-fitting as indicated by the distance between the training and test error, indicating that if we trained with more individuals (i.e. collected more celebrities) we may be able to increase performance even further. The over-fitting is the results of the large number of parameters in the projection matrix Φ (5000) and the rather limited set of individuals we train with (we train with up to 85 individuals and a total of about 150 images).

In Figure 9 we compare the performance of various feature sets (4 of them) using both L_1/L_2 distances. We note that using Gradient features yielded the high performance in Figure 6, i.e. prior to mapping. While in our experiments using the mapping Φ the intensity yielded the best performance. This is most likely due to the large size of the gradient feature vectors used compared to the size of the feature vectors used with only intensity (the intensity patches extracted were smaller than the gradient patches extracted). Indeed if we analyze the variance of the PCA coefficients obtained when projecting to 100 dimensions, we find that the gradient vectors encompass about 90% of the variance while the intensity vectors encompass about 65% of the variance. The plots in Figure 9 indicate that this loss of information has detrimental effects on performance. Note that due to over-fitting, increasing the projected space of PCA dimensions above 100 results in worse performance as well. I.e. if the number of parameters which must be optimized in our map Φ, we will suffer from over-fitting. In Table 2 we show which mappings perform best on all three performance metrics. We also note the large performance gains achieved over not using the mapping, i.e. using only the extracted feature vector $\vec{x}$ in the variance performance metrics.

Cluster Distance			Best Rank			Top 25 Precision		
Rand	Best	Feat	Rand	Best	Feat	Rand	Best	Feat
.5	**.13** (.25)	L1 I 9	19.2	**2.7** (4.2)	L2 I 9	1.92	**5.8** (4.48)	L1 I 7

Table 2. The best mapping functions. Rand: performance if we chose random images. Best: the performance of our best mapping algorithm. In parentheses the performance without mapping, i.e. on the raw feature vectors. Feat: the feature set used. L1/L2: the distance metric used. I: intensity features. 7/9 the size of the patches used. E.g. 7: 7×7 patches.

Figure 10. (Left) The raw returned results for Google search for Nick Lachey. Index of results goes from left to right and top to bottom. (Middle) Results after our filtering and unsupervised clustering. Note that the results are significantly more consistent. (Right) X-axis: index of returned image. Y-axis: precision of images returned so far. Precision is measured as the percentage of images in the set returned so far which have been labeled as 'Good'. Each line represents performance when a different set of filters has been applied to the results. From top to bottom: (Google Raw): the raw results returned returned by Google. (VJ Filter) Only images which have at least one Viola and Jones detection. (Single Det) Filtered by results which have only a single VJ detection which passes our confidence measure. (Single Det, Group) Perform clustering on the returned results. (Single Det, Group, Map) First map the results using our perceptual map, and then cluster them. The latter tends to perform the best across all 20 celebrity searches. Note that the blue line (raw results) spans up to 250 returned images, while the green line spans to roughly 100 images as roughly 150 images have been disregarded by our filters.

7. Image Search: Unsupervised Clustering

We now consider how to use our system to automatically increase the precision of returned images from a Google image search for a particular celebrity. Our data-set consisted of the first 250 returned images from Google image searches for 20 different celebrities. Naturally, the results returned by Google do not contain only the celebrity of interest, but rather a grab-bag of images loosely related to the celebrity query. In order to evaluate the quality of each returned image from our searches we used the following criteria: (1: Good) The image contains the celebrity of interest in a roughly frontal view, $+/-20$ degrees from frontal. (2: Ok) The image contains the celebrity in a strange pose or under heavy occlusion. (3: Bad) The image does not contain the celebrity of interest. Our task then translates to filtering these returned images from searches in order to have the highest percentage of good images in the first results returned.

We proceed as follows to increase the performance of Google searches for an invidual: (1) Run the Viola and Jones (VJ) [10] face-detector and filter out all images which do not one face detection and which do not pass our confidence measure as described in 4. (2) Project all remaining faces to a face space created using 85 different celebrities as described in Section 6. (3) Cluster all remaining faces. How do we cluster? Consider that we have N available images. We seed our clustering algorithm by finding the set of K images which have the smallest average distance between them. We then proceed by greedily adding new images to the cluster that have the minimum average distance between all the images already contained in the cluster. This method, although simple, produces good results in practice. We typically set $K = 3$ for our experiments and N is typically in the range of 50-100. We note some changes in performance when setting $K = 2$ although the qualitative results when viewing the returned images look comparable.

We compare results for a specific celebrity search (Nick Lachey) using our system in Figure 10 and across all 20 celebrity searches in Figure 11 and note that: (1) our system dramatically outperforms the raw Google celebrity search as evidenced by the difference between the red and green lines, and (2) our learned mapping provides a large per-

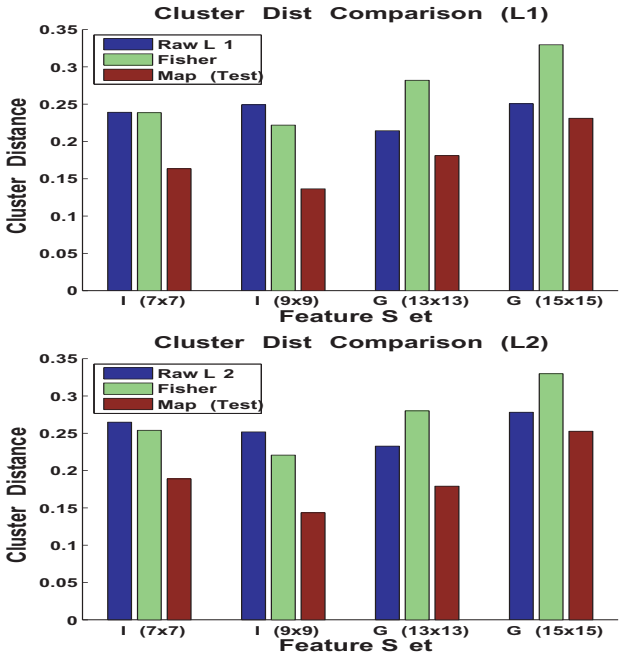

Figure 9. Comparison of the Cluster distance performance metric. X-axis is 1 of 4 different feature representations: 7×7 Intensity patches, 9×9 Intensity patches, 13×13 Gradient patches, and 15×15 Gradient patches. These were chosen as they performed the best in Figure 6. First column is the raw L1 distance performance before learning the mapping. Second column is the performance using FisherFaces. Last columns is the performance using our mapping. We outperform the other metrics in every feature set tried here. Notably gradients perform worse here relative to intensities, see text for a discussion. (Bottom) Same plot as above but using the L2 distance to optimize the mapping and to compute the raw distance. The same trends seem to hold. $\alpha = 1$ for these experiments. Results averaged over 3 iterations.

formance boost over not using a mapping as indicated by the difference purple and green lines. Note that this system makes use of supervision only when learning the generic mapping function for faces.

8. Discussion

There are numerous topics for further exploration, including using additional features such as the hair (which has been shown to perform remarkably well by [7] in which they describe experiments confusing Al Gore for Bill Clinton by mapping the hair from one to the other). In addition, the work could be extended to encompass larger variations in pose by learning a more powerful distance metric. Finally we would like to deploy and measure the performance of our system on larger data-sets.

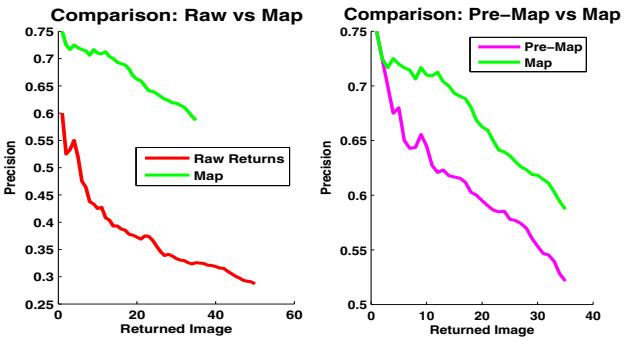

Figure 11. Unsupervised clustering: average performance over 20 celebrity searches (see Figure 10 for a description of the axes. (Left) Comparison of raw returned results and mapped returned results using same metric as Figure 10. Our method dramatically outperforms the raw returned results from Google. (Right) Comparison of clustering with and without mapping. We achieve significant gains by learning a generic face mapping function.

References

[1] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *PAMI*, 19(7):711–720, 1997.

[2] T. L. Berg and et al. Names and faces in the news. 2004.

[3] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. In *BMVC*, 2006.

[4] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. In *FG*, pages 441–448, 2006.

[5] Intel. Opencv computer vision library. In *http://www.intel.com/technology/computing/opencv/*.

[6] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database. *PAMI*, 25(12):1615–1618, 2003.

[7] P. Sinha. Identifying perceptually significant features for recognizing faces. In *SPIE*, volume 4662, pages 12–21, 2002.

[8] P. Sinha and et al. Face recognition by humans: nineteen results all computer vision researchers should know about. In *Proceedings of the IEEE*, Vol 94 N2, 2006.

[9] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *ECCV*, May 2004.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[11] E. Xing, A. Ng, M. Jordan, and S. Russel. Distance metric learning with application to clustering with side information. In *NIPS*, 2002.

[12] L. Yang. Distance metric learning: a comprehensive survey. In *Michigan State Universiy*, 2006.